

**Supporting Pet-to-Family Reunification in Disaster by
Leveraging Human and Machine Computation**

by

Joshua Franklin Barron

B.S., New Mexico Institute of Mining and Technology, 2010

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Master of Science
Department of Computer Science

2013

This thesis entitled:
Supporting Pet-to-Family Reunification in Disaster by Leveraging Human and Machine
Computation
written by Joshua Franklin Barron
has been approved for the Department of Computer Science

Leysia Palen

Kenneth Anderson

James Martin

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Barron, Joshua Franklin (M.S., Computer Science)

Supporting Pet-to-Family Reunification in Disaster by Leveraging Human and Machine Computation

Thesis directed by Prof. Leysia Palen

Displacement of pets in and after disaster events is a serious matter to families and creates public safety and health issues at large. During and after Hurricane Katrina, an estimated 200,000 pets were displaced from their guardians, with only five percent of these pets ever being reunited with their guardians. One of the major obstacles to successful pet-to-family reunification is the large search space individuals must navigate in search of lost pets. Grounded in the theory of crisis informatics, which studies how people use information communication technology (ICT), this work leverages the phenomenon of digital volunteerism to better address the problem of pet-to-family reunification.

This effort culminates in the form of **No Place Like Home**, which is an online platform for use by digital volunteers interesting in aiding pet-to-family reunification efforts in a disaster. This work describes this system in terms of its design features and software architecture. The primary design features of the system are an accessible user interface, collaboration mechanisms (in the form of social network support and chat rooms), and social capital. The other key feature of this system is how it incorporates human computation (performed by the crowd of digital volunteers) with machine computation (performed by information retrieval and classification systems) in a collaborative manner which plays to the strengths of each type of computation.

An evaluation of the machine learning components of the system in an experimental setting reveals that the positioning of machine computation as supporting the larger activities of digital volunteers is promising. With this validation, the system is poised for future public deployment in a disaster scenario and further study.

Dedication

This work is dedicated to Genevieve, Melanie, and Peter.

Acknowledgements

I would like to thank my committee members for their assistance. Special thanks must be given to my advisor, Leysia Palen, for her consistent and valuable guidance regarding this work. I would also like to acknowledge the members of Project EPIC for their assistance, but especially Mario Barrenechea and Joanne White for their work with me on the design of **No Place Like Home**.

Contents

Chapter		
1	Introduction	1
2	Background and Approach	3
2.1	Pets in Crisis Situations	3
2.2	Digital Volunteerism	4
2.3	The Data Scouts Framework	6
2.4	Machine Learning to Support Human Activity	7
2.5	Envisioning a System for Pet-to-family Reunification	8
3	System Design	11
3.1	Application Features and Interface Design	11
3.1.1	Accessible User Interface	12
3.1.2	Collaboration and Social Capital	16
3.2	Software Architecture	19
3.2.1	Artifacts and Data Management	19
3.2.2	Implementation	22
4	Machine Learning System Design and Evaluation	32
4.1	Design Considerations	33
4.2	Lucene Index	34

4.3	Pet Classifier Architecture	35
4.4	Match Classifier Architecture	36
4.5	Combined Scoring System	37
4.6	Experimentation and Evaluation	38
4.6.1	Corpus	39
4.6.2	Classifier Implementation	40
4.6.3	Pet Classifier Training	40
4.6.4	Pet Classifier Evaluation	42
4.6.5	Match Classifier Training	42
4.6.6	Match Classifier Evaluation	44
4.6.7	Combined Evaluation	44
5	Conclusion	49
5.1	Summary	49
5.2	Reflection and Contributions	51
5.3	Future Work	51
	Bibliography	53

Tables

Table

3.1	Tasks facilitated by the system	11
3.2	Artifacts	20
3.3	Views provided by authentication subsystem	28
3.4	Translations performed client-side by chat subsystem	28
3.5	Views provided by chat subsystem	29
3.6	Views provided by reporting subsystem	29
3.7	Views provided by matching subsystem	30
3.8	Public interface for Machine Learning subsystem	31
4.1	Variations of the scoring formula	47
4.2	Classifier implementations	47
4.3	Pet classifier evaluation	48
4.4	Match classifier evaluation	48
4.5	Combined evaluation	48

Figures

Figure

2.1	Data Scouts Roles	6
3.1	First Interface Prototype	14
3.2	Second Interface Prototype	15
3.3	Third Interface Prototype: selecting a pet to work on	17
3.4	Fourth Interface Prototype: finding a match for a pet	18
3.5	Database Design	21
3.6	Deployment Diagram	23
4.1	Sample Pet in Corpus	39
4.2	Pet classifier Annotation Interface	41
4.3	Match classifier Annotation Interface	43

Chapter 1

Introduction

3 4

$\frac{2}{1} X \frac{5}{6}$

8 7

Disaster events observed in recent years have been characterized by a strong element of digital, public participation. While public involvement in disaster events is not a new phenomenon, information communication technology (ICT) [16] makes this involvement more visible and enables new types of volunteer activities for interested members of the public. The field of crisis informatics [15] is the study of how people use ICT in crisis; it examines socio-technical issues in emergency response, and considers the actions of both official responders as well as members of the public. The actions of members of the public during disaster, and including the actions of digital volunteers [20] in disaster have been the focus of recent crisis informatics research.

The emergence of interested citizens participating in a form of remote volunteerism during a disaster (also known as digital volunteerism) presents an interesting opportunity to develop software systems which support these volunteers' work and to explore issues at the intersection of human and machine computation. The purpose of this work is to describe one such system whose purpose is to utilize digital volunteers to facilitate pet-to-family reunification in disaster. This system is built to incorporate and synthesize human computation, crowd work, and machine learning; this thesis is an exploration of various design, implementation, and evaluation issues that arise in the course of creating such a system. In addition to describing the design and architecture of this system, this work presents an experimental evaluation of the system's machine learning component, which is

designed to support and assist the activities of digital volunteers.

The remainder of this work is divided into four chapters. The second chapter discusses related literature and the problem of pet-to-family reunification in disaster. The third chapter presents the interface and system design of **No Place Like Home**, a software system which facilitates pet-to-family reunification. The fourth chapter details the machine learning components of this system and presents an evaluation of this system in an experimental setting. The fifth chapter concludes this work by reviewing key ideas and findings in each section, reflecting on the work as a whole, and presenting directions for future research.

Chapter 2

Background and Approach

This chapter examines ICT-supported citizen participation in crisis events (specifically the phenomenon known as digital volunteerism) and how it relates to the problem of pet-to-family reunification in disaster. In addition, this chapter discusses the need for computational support of volunteers' activities in the form of machine learning and examines literature that has influenced my design of this system.

2.1 Pets in Crisis Situations

Though exact figures are difficult to obtain, it is estimated that 50 to 70 percent of American households (approximately 61 to 86 million) include at least one pet. When disaster strikes, pets are often left behind in evacuations because they cannot be transported or allowed into shelters. In the case of large-scale disasters, the displacement of pets creates a variety of serious problems. During Hurricane Katrina, an estimated 200,000 pets were displaced from their guardians; only five percent of these pets were ever reunited with their guardians [9]. Many displaced pets were disposed of or “rehomed” to families in other states, which in some cases even resulted in legal disputes over pet ownership when the original guardians of a pet located it [13].

Animals displaced by disaster can create large public health hazards. In addition, displaced or abandoned pets can be the source of public safety issues as evacuated disaster victims will return to hazardous areas in order to retrieve them. The challenges of incorporating pets into disaster planning were recognized by the federal government's passing of The Pets Evacuation and

Transportation Standards (PETS) Act of 2006 in response to challenges faced during Hurricane Katrina and its aftermath.

Furthermore, disaster victims who leave pets behind may be ill-suited to attempt to locate them remotely; in the aftermath of a disaster, victims (especially those residing at a temporary shelter) may have limited access to communications technology such as phones or computers. This was seen in the conditions of Hurricane Katrina victims who were located at shelters - these residents had extremely limited access to land line phones and although some residents had cell phones, coverage was limited and charging them was a challenge [16]. It is also unrealistic to expect disaster victims to spend what little time they have with ICT in the aftermath of a disaster searching for lost pets. Finally, a single person may be unable to adequately search for a given pet in a potentially vast search space. For example, if even ten percent of lost and found pets in Hurricane Katrina were reported digitally, an individual would need to search through approximately 20,000 pet reports to find a single animal. Directing the efforts of the crowd (in the sense of crowd work) can mitigate this problem by increasing collaboration and distributing work.

2.2 Digital Volunteerism

An interesting development in the field of crisis informatics has been the study of digital volunteerism. Digital volunteerism [20] is a technological expansion of traditional volunteer activities that occur in disaster situations [5, 16], and is characterized by improvisation and self-organization by the people involved in volunteer activities. Digital volunteers are defined by their location, as they are usually volunteers who are geographically removed from a disaster but still want to be involved in volunteer efforts via online activity.

Starbird and Palen explored digital volunteerism in the aftermath of the 2010 Haiti earthquake as remote volunteers converged through Twitter to process information and make connections between requests for help and offers of assistance [20]. Self-named “voluntweeters”, these digital volunteers primarily engaged translating crisis tweets from their original format into the “Tweak the Tweet” syntax [21], which is machine readable, effectively synthesizing unstructured information

(from multiple sources) into structured, usable data. Many interviewed volunteers engaged in auxiliary tasks in addition to the core translation activity, such as **verifying** the content of a tweet or acting as a remote operator by **routing** information [20]. It is important to note that the volunteers in this study performing these more complex tasks were unable to use the core social media platform (Twitter) to perform the entirety of their work; this suggests that there is space for specialized tools to be built that leverage social media but accommodate the needs of digital volunteers.

Specialized tools can not only accommodate the needs of digital volunteers but also deal with issues of information convergence that arise when dealing with crowd work in the crisis space. Starbird identifies these issues [19] as:

- **Noisy information space:** the information space of a social media platform such as Twitter, even when filtered through the use of mechanisms such as hashtags, is still noisy and must be filtered further so that relevant information can be identified.
- **Loss of context:** information which is routed through social media is subject to modification by the users performing the routing mechanism (in the case of Twitter, this often takes the form of the retweet mechanism). Modification of information and even the simple act of information passing can cause loss of contextual information.
- **Misinformation and disinformation:** incorrect information can hinder the activities of digital volunteers and even be dangerous in a disaster situation; information must therefore be subject to a verification mechanism.
- **Organization and data structure:** to be more useful, information needs to be compatible with computational resources, such as search engines and databases. Information must therefore be structured in a systematic manner.

The use of specialized tools in the disaster space can alleviate these issues in a variety of ways. First, specialized tools will most likely impose some structure on the the specific data being

used within a system; this immediately addresses the problem of organization and data structure of information. Furthermore, because a specialized tool is not limited by structural limitations on data format present in some major social media platforms (such as Twitter), it can more effectively address the problem of loss of context. For example, retweeted information on Twitter may lose contextual information due to user modification or simple loss of textual content stemming from the 140-character limit on tweet size. A specialized tool might overcome this limitation by storing more detailed history of a particular information item, thereby preserving the various contexts associated with that item. Finally, a specialized tool can integrate computational and social solutions to address the problems of noisy information and misinformation, which might include, for example, an information retrieval system catered to the data structures managed by the tool to alleviate noise and including a voting mechanism to mitigate misinformation.

2.3 The Data Scouts Framework

The goal of the Data Scouts framework [1, 14] is to classify the digital volunteer into one of three roles: the **data scout**, **consolidator**, or **checker**. The characteristics of these roles are briefly described in Figure 2.1.

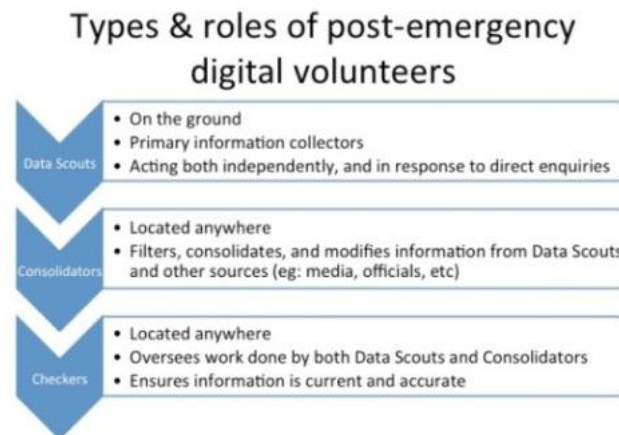


Figure 2.1: The three types of digital volunteer activity in the Data Scouts framework

This framework introduces a new category of volunteer, the data scout, whose role is to provide on-the-ground data for use by remote digital volunteers. In the domain of pets in disaster, the data scout is the volunteer who is at a shelter reporting animals that reside there, or just an individual who has lost or found a pet during the disaster.

The importance of this framework with regard to this work is that it classifies the behaviors of digital volunteers discussed previously into specific roles. Consolidators engage in synthesis tasks like information translation and routing; checkers engage in the important task of information verification. It is important to note that the roles delineated by this framework are not by any means mutually exclusive; it is highly likely that digital volunteers might take on one or more roles during a crisis event. Nevertheless, the roles that this framework provides are highly useful because we can use them as a starting point for system design. The system described in this work makes heavy use of the roles described here.

2.4 Machine Learning to Support Human Activity

Machine learning is often thought of as being orthogonal to human computation, but there are many machine learning approaches and paradigms that view human computation as complementary to machine computation; this work embraces this view. The idea that human and machine computation are at their best when placed in complementary roles is not a new one; in 1960, J. R. Licklider envisioned the future of computing as one where man and machine acted in symbiosis, with computers acting as colleagues to their human counterparts [7]. This vision is reflected in many machine learning systems today, such as recommendation systems and autonomous agents.

The study of autonomous agents has a heavy influence on the design of the machine learning components presented in this work. Autonomous agents encompass a variety of artificial intelligence techniques and have a somewhat controversial definition [4], but the essential idea is one of a learning software system which acts in collaboration with a human user. Such a system necessarily is viewed as an assistant to the user, and this assistant is generally expected to become more proficient over time at whatever tasks it is delegated. Agents can assist users in a wide variety of ways, ranging

from performing complex computation, training users, performing tasks on behalf of users, or even helping multiple human users collaborate [8, 11]. As such, the set of problem domains in which they are applicable is large and includes information filtering and retrieval, which is the core task of the machine learning component presented in this work.

A key aspect of autonomous agents is that in collaborating with users, they are not the primary authoritative actor that makes decisions. Agents may perform tasks on behalf of users or recommend courses of action, but they do not prohibit users from taking independent action. This quality helps mitigate the imperfect nature of artificially intelligent systems and can focus the design of machine learning systems onto tasks which are ideally suited to their use.

Placing machine computation in the role of a collaborator instead of a complete solution to problems is a view that meshes well with many problems in crisis informatics. Machine learning can play a large role in crisis informatics technology, especially in information retrieval and extraction. For example, natural language processing and classification techniques have been shown to be effective in deciding whether or not a tweet contributes to **situational awareness** in disaster [22]. At the same time, past research details the unique challenges of applying machine learning algorithms to the types of information common in the crisis space [2, 20]. Because we cannot expect machine learning to be a silver bullet solution, especially in the space of crisis informatics, this work takes the approach of implementing machine learning solutions that solve tasks unsuited for human computation. Furthermore, the machine learning components are designed from an agential paradigm and take opportunities to learn from human computation to better assist human activity.

2.5 Envisioning a System for Pet-to-family Reunification

The problem of pet-to-family reunification is an ideal context in which to build a system which utilizes digital volunteers and combines human and machine computation. As this chapter has discussed, pet owners who lose their pets may be ill-suited to finding them on their own. By utilizing digital volunteers (as defined by the Data Scouts framework), we can focus the efforts of an enthusiastic user population on a task that is an important societal problem. This section examines

some of the challenges faced in pet-to-family reunification and discusses how these challenges inform the design of a system, for use by digital volunteers, to solve this problem.

The problem of pet-to-family reunification is complicated because of the fact that pets cannot self-identify after being found in the wake of a disaster by an individual or organization. Furthermore, the search space of all pets reported lost and found to online services during a disaster is both at once large and sparse — there may be thousands of reported pets but there is still only a small chance of finding a correct match for any given pet in the space. Textual profiles of pets generally do not contain sufficient information to be the sole resource used in identifying a pet. Visual information, such as pictures of a pet, are likely the most valuable identifying information contained in a profile of a lost or found pet. However, computational techniques for processing this information are likely to be ineffective within the disaster domain. For example, a pet owner may submit an older photo of a pet when reporting that pet as lost, but a data scout in a shelter may include a photo of that pet in a report that has different visual context, i.e., environment, the age of the pet, the angle of the photograph, the inclusion of other pets in the picture, and other factors. Nevertheless, I posit that this visual matching task is something that humans, who are highly capable of overcoming challenges like noise and ambiguity, are very much suited to completing.

Using human computation to solve the problem of visual matching is good approach to tackling the problem of pet-to-family reunification, but it does not solve the problems of scale mentioned above. By designing a software solution around digital volunteerism, we can expand, in a sense, the quantities of human computation available to tackle the problem. By incorporating social and collaborative features into this solution, we can improve the quality of human computation by providing mechanisms for information routing and verification. However, even by expanding our user base from pet guardians (searching, in general, for a single lost pet) to cadres of digital volunteers, we cannot expect users to scour over thousands of pet reports. This is where machine learning systems can play an important role in assisting digital volunteers attempting to reunite lost pets with their guardians.

Although machine learning cannot reliably be used to pick out matches between reports of lost

and found pets, if we consider lost and found pets as documents, we might design an information retrieval system to assist volunteers in retrieving what appear to be the most relevant reports for any given pet. Textual features, as mentioned above, may be insufficient to select a single match out of the entire search space, but it is likely that textual features might be useful in selecting a range of potential matches for consideration by volunteers. The machine learning component of the system is engaged in information filtering on behalf of digital volunteers.

Furthermore, if we design the machine learning component to take an agential stance, we must look at the activities of digital volunteers (human computation) as opportunities for the machine learning system to improve itself. This work explores two such training activities: volunteers editing pet reports to improve their usefulness and volunteers proposing and recommending matches between a lost pet report and a found pet report (the fourth chapter presents an evaluation of the effectiveness of this approach in an experimental setting). By taking this approach, we have proposed a system in which human computation is used to inform machine computation, which in turn provides greater assistance to future human computation.

This chapter has demonstrated that displaced and lost pets in disaster create a variety of health and safety problems to individuals and the public. This chapter has also examined the phenomenon of digital volunteerism within the context of crisis informatics, as well as the role machine learning systems can play in this space. The system proposed in this section takes the form of **No Place Like Home**, an online matchmaking tool which acts as an infrastructure supporting digital volunteers in facilitating pet-to-family reunification. This system’s interface design and conceptual framework was initially explored in Barrenechea et al. [1]. The design and architecture of this system is presented in the following chapter.

Chapter 3

System Design

This chapter presents the design and architecture of **No Place Like Home**, an online application that serves as a platform for facilitating pet-to-family reunification through the activities of digital volunteers. The beginning of this chapter discusses the various interaction design requirements we¹ generated during prototyping this system and their relationship to other system components. The following sections are focused on detailing the software architecture of the system.

3.1 Application Features and Interface Design

No Place Like Home is a system for use by digital volunteers. Examining the activities of digital volunteers through the lens of the Data Scouts framework allows us to elicit system requirements. Table 3.1 lists the primary tasks we envisioned being facilitated by the system.

Table 3.1: This table shows the most prominent tasks facilitated by the system, divided by the role of the user performing them.

Data Scouts Role	Tasks
Data Scout	Reporting lost or found pets.
Consolidator	Matching lost pet reports with found pet reports.
Checker	Verifying matches proposed by consolidators, editing pet reports.

Of these tasks, the matching task facing the consolidators who use the system is the most

¹ I use the pronouns “we” or “us” deliberately in this chapter to refer to work done in collaboration with Mario Barrenechea and Joanne White [1].

complex to design for, and it is where we focused most of our efforts. The matching task is difficult for a variety of reasons. There are technical challenges to overcome, the most difficult of which is that the size of the search space for possible matches is very large; this is the motivating factor behind the inclusion of machine learning components in the system. Even with assistance from these components, users of the system will still need to sift through several candidates before completing the matching task. Finally, because the machine learning components act in collaboration with users (and because we cannot expect them to be perfectly accurate in completing their tasks), we must provide tools for users to manually complete the matching task.

There are also several social challenges that have been addressed during the design of this system. Many of these problems are related to user demographics; although consolidators all engage in the same task, they are likely to be a highly diverse group in terms of age, background, enthusiasm, and technical experience. In addition, because collaboration between volunteers is a key component to the completion of virtually all the tasks facilitated by the system, the system must provide a collaboration mechanism and a social networking system.

The solutions chosen to address these challenges motivate the key features of **No Place Like Home**, each of which are discussed individually in the following sections.

3.1.1 Accessible User Interface

Because of the diversity of the projected users for **No Place Like Home**, our design solution must be accessible to users in a variety of ways. Part of our initial design work was in generating personas to model the users of the system: these personas included a wide range of profiles, ranging from elementary students to young adults to senior citizens[1]. As such, the design of our user interfaces is motivated by these goals:

- The default work flow to complete tasks, especially the matching task, must be intuitive and simple to complete.
- The tool must provide alternate work flows to accommodate the different levels technical

expertise and motivation that users may possess; this is to say that the depth and complexity of the interaction available to users must vary based on these factors.





In pursuing these design goals we have created several interfaces in an iterative manner. Our prototype interfaces have all been focused on the work flow for the matching task. The design changes made from iteration to iteration have been informed by user centered design techniques (generally cognitive walkthroughs and “thinking aloud” exercises with users). Figures 3.1 and 3.2 show the evolution of the interface design from low to higher fidelity prototypes.

The design of the user interfaces for **No Place Like Home** is an ongoing process. After completing our initial prototyping [1] and user testing, we discovered problems with the initial design of the matching interface:

- Virtually all participants ignored the advanced search capabilities, instead favoring visually browsing pet reports.
- Many participants needed to temporarily store pet reports (that they considered match candidates) while considering other pet reports for a match.
- The two-column design (see Figure 3.2) was not optimal for the matching task, as participants often just picked a pet report from one side (indeed, our testing scenarios were even set up this way) and then, of course, completely ignored that column for the duration of the task. This meant that valuable screen real estate was going to waste in addition to being a source of confusion for users.

Solving these problems is the subject of our ongoing design work. We have labored to separate the interfaces for the matching task so that they more closely match the observed work flow of testing participants; we now have a separate interface for selecting a pet to “work” on and another interface for finding a match for that pet. The new matching interface is much more visually oriented, drawing design inspiration from sites such as Pinterest, and incorporates new functionality

Project EPIC PetMatch

Filtering Options ↑

Search Terms:

Contact Info:

Report Date: to

Location: within

Show ☒ Top Potential Matches ☐ Newly Listed

Lost

Bob the Dog +

golden retriever small puppy ✖

Sally the Dog +

golden retriever small puppy ✖

BillyJoe +

golden retriever small puppy ✖

Found

Dog "Bob" +

golden retriever small puppy ✖

Dog "Bob" +

golden retriever small puppy ✖

View More Potential Matches
Suggest Match

Match Feed

[Mario has proposed a match.](#)

[Bob the Dog and Dog "Bob" were matched!](#)

✖

✖

[Sally Cat and Lost Tabby were matched!](#)

✖

✖

Everyone
FindingDogs
BoulderDataScouts
System Feed

Josh > any luck on that golden retriever yet?

Jo > @Josh, maybe [#1140 ?](#)

[Mario has proposed a match. Click here to view and confirm this match.](#)

SYSTEM > Josh, Jo, and Leysia have confirmed Mario's match. All participants have earned 5 super pet points!

Josh

Jo

Mario

Steve

Leysia

Ken

Kate

Type a message here...

Figure 3.1: Low-fidelity mockup of an interface to complete the matching task.

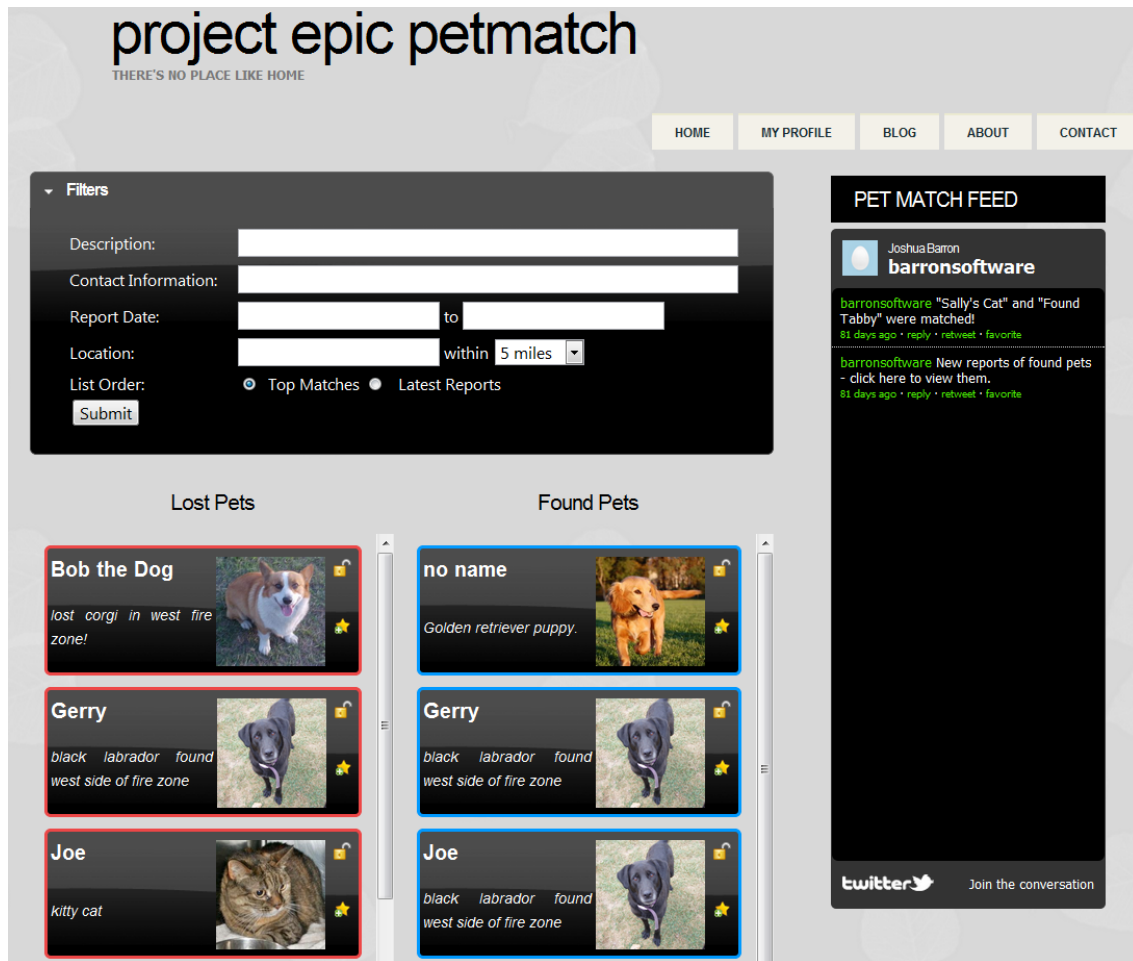


Figure 3.2: Medium-fidelity mockup of an interface to complete the matching task.

to allow users to preserve their work in the form of a temporary workspace which holds candidates that the user has selected. Figures 3.3 and 3.4 display early prototypes of these new interfaces.

3.1.2 Collaboration and Social Capital

Collaboration and networking are key to facilitating the crowd work performed by digital volunteers. Because digital volunteers have been shown to connect to each other both via pre-existing connections as well as connections which emerge around the completion of a task [20], this social networking mechanism should honor existing social connections as well as those that are forged during an event. **No Place Like Home** does this by allowing users to access the system via their Facebook or Twitter account; alternatively, the user can also rapidly create a site-specific account. In addition to providing a quick authentication mechanism, the system can also use this information to import the user's existing social connections.

Supporting social networks provides a backbone to the social features of the system, but an active mechanism to support collaboration is still needed. **No Place Like Home** therefore implements a chat room functionality that lives at the bottom of every interface in the application. Chat rooms allow users to connect with their social network and coordinate as they complete tasks within the system. Chat rooms are automatically created for groups of people working on specific pet reports, and the chat interface is present at the bottom of every user interface in the system. Logs of chat activity for these rooms are attached to the corresponding pet report and saved as they represent valuable contextual information for future work with that pet.

Chat rooms are also designed to support contextual links (i.e., a user can easily insert a link to an artifact, such as a pet report, into a chat message) to facilitate quicker collaboration. In addition, automated messages are sent to relevant users when certain artifacts are created within the system - for example, a user might be prompted to vote on a match proposed by one of his or her friends. This is another key feature of the system which facilitates rapid verification of pet matches by on-task populations of users.

In order to provide some regulation to the community activity which takes place during an

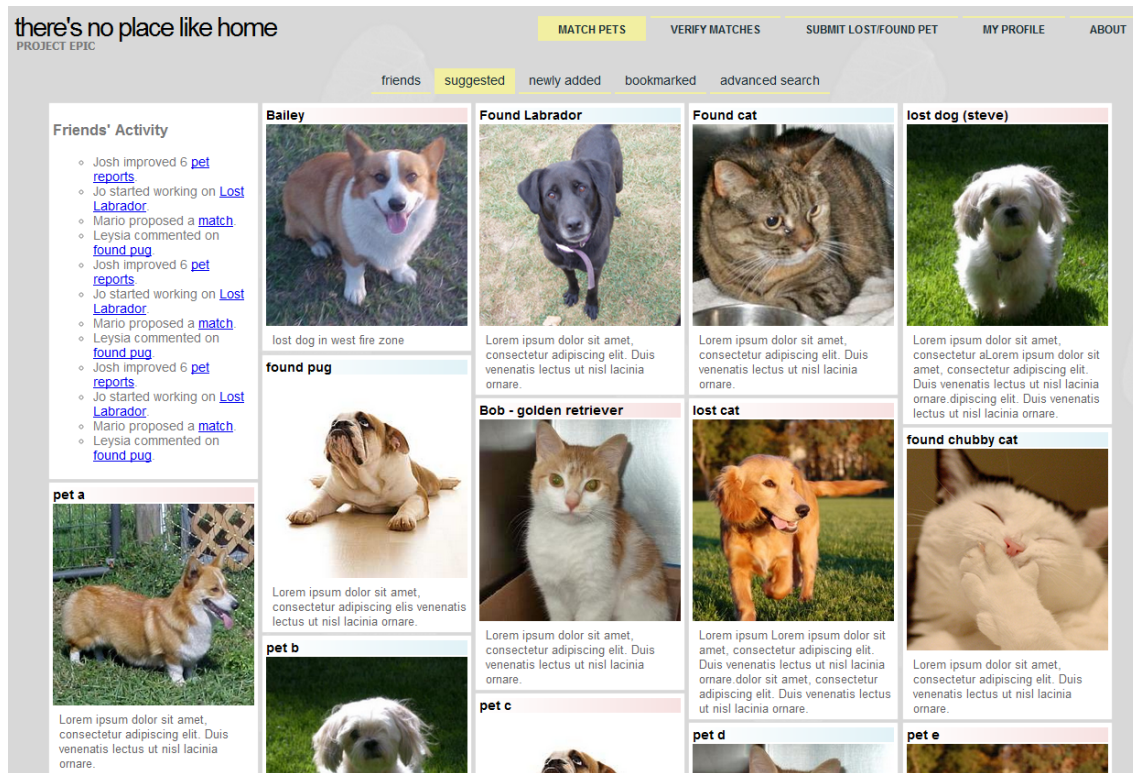


Figure 3.3: Medium-fidelity mockup of the pet selection interface; a user selects a pet to work on the matching activity via this interface.

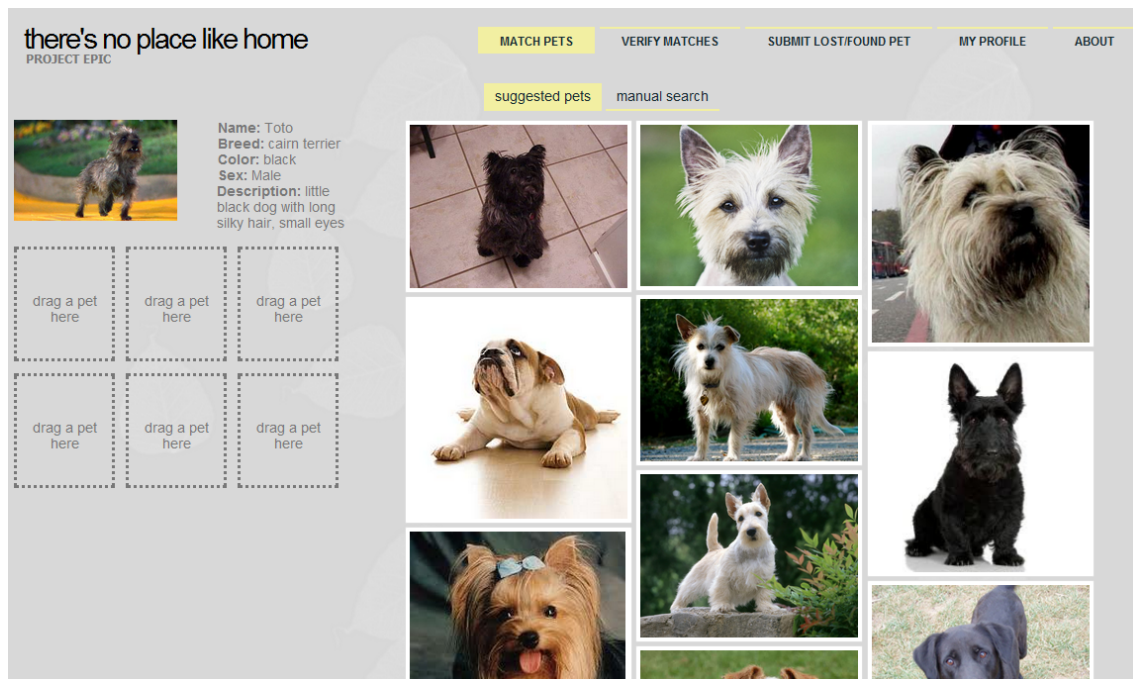


Figure 3.4: Medium-fidelity mockup of an interface to complete the matching task, utilizing a new workspace area (for drag-and-drop actions) and increased screen real estate.

event via the system, as well as to provide an incentive mechanism and a potentially interesting feature for the machine learning components of the system, a social capital mechanism is also present within the system. Users earn social capital (represented as a numerical value) for completing various actions within the system, such as submitting a pet report, editing a pet report, proposing a match for a pet report, or voting on a proposed match for a pet report. Furthermore, if a user is the creator of an artifact within the system (such as a pet report or a proposed match), the social capital of that user can be affected when other users take actions on that artifact (i.e., voting on a proposed match). Social capital is a public indicator of the status of an individual within the community, and in addition, the system makes use of it to automatically grant administrative capabilities to the highest rated users.

3.2 Software Architecture

I have discussed the system design and key features of **No Place Like Home** in the previous sections of this chapter. This portion of the chapter discusses the architectural design and implementation of the software which makes up **No Place Like Home**. The system is implemented primarily as a web application under the **Django** framework. Data is managed via document collections in a **MongoDB** store.

The following sections discuss the various data artifacts managed by the system, the deployment of the system, and the implementation of the Django applications and other Python components which make up the system.

3.2.1 Artifacts and Data Management

Although the interactions facilitated by the system are rich and complex, the data artifacts generated by users and managed by the system are relatively simple. Artifacts are generally related to either users and user activity or pet reports. Table 3.2 describes the various artifacts managed by the system.

Instead of storing data in a relational database, which is the typical design choice in client-

Table 3.2: This table details the artifacts tracked by the system.

Artifact	Generator	Description
Pet	Data Scout	Represents a lost or found pet reported to the system.
Match	Consolidator	Describes a proposed match between one lost pet and one found pet.
User	Any	Represents a user of the system and tracks work items in addition to authentication information.
Chat	Any	Represents a chat room in the system - can be linked to a specific Pet .

server applications, **No Place Like Home** makes use of MongoDB², a document-oriented data store. The reasons for this design decision are detailed in the following section.

Utilizing a document-oriented data store allows the artifacts tracked by the system to be stored in a format which is very similar to their conceptual structure. Documents can aggregate complex data structures without resorting to linking, which leads to a design that is fairly concise. The decision to make an artifact or its properties a top-level document in the collection usually comes down to whether or not doing so would prevent data replication rather than concerns about normalization and performance that are common in relational designs. The design of a document as an artifact's representation is akin to doing object-oriented design; so much, in fact, that I choose to detail the implementation of the system's database as a class diagram in Figure 3.5.

Data Management

The system makes use of the document-oriented data store, MongoDB, to manage the aforementioned artifacts. The choice to use a document-oriented database over a traditional relational database was made for several reasons. Document-oriented databases are capable of representing complex hierarchical structures in a simple manner, allowing data artifacts to be implemented in the database in a very natural and logical manner. Also, document-oriented

² <http://www.mongodb.org/>

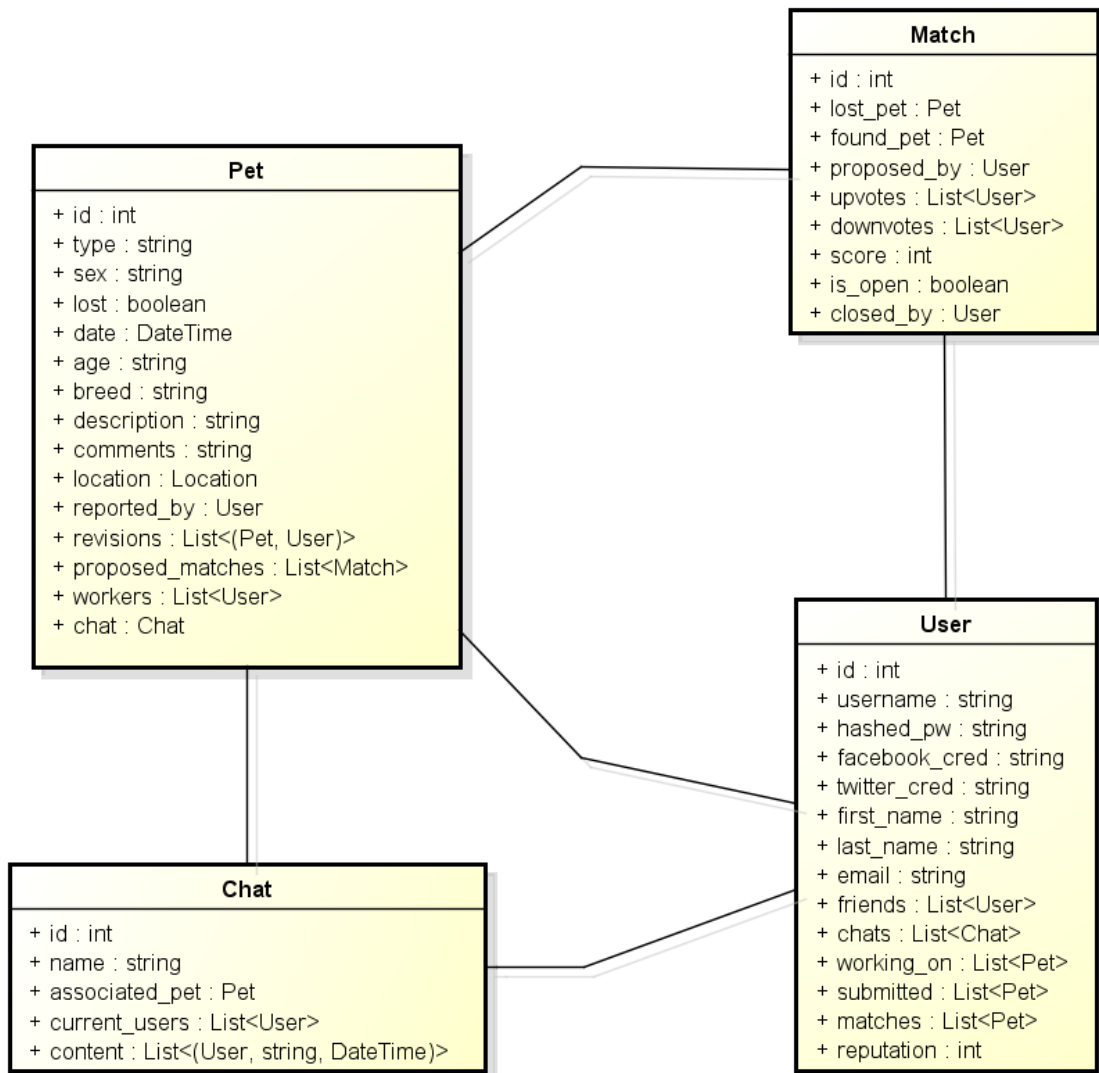


Figure 3.5: Object-oriented database design for **No Place Like Home**. Note that there are several associations between several of the artifacts, and that this diagram does not detail each association; refer to the contents of each artifact for more detail.

databases such as MongoDB typically exhibit much higher scalability than relational databases, which could be a essential to system performance in a large scale deployment.

Development considerations also influenced the choice of MongoDB as the database system for **No Place Like Home**. Because the database is in some sense “schema-less” in that documents in a collection do not need to share common formats, the specification of the artifacts tracked by the system was able to evolve naturally and in tandem with other development. If a traditional relational database management system had been used, the uncertainty of the data specification that existed at the outset of development would have increased overall development time because of the increased complexity of data specification in a relational format and the difficulty of making changes to an established schema. Finally, MongoDB was chosen as the database backend over other document-oriented databases in particular for its ease of use with Django, the web application framework used to implement **No Place Like Home**.

3.2.2 Implementation

No Place Like Home is implemented primarily as a series of Django applications. Django³ is a leading Python framework for creating dynamic web applications using the Model-View-Controller pattern[3], although Django uses its own terminology, calling the controller the “view” and the view the “template”.

Django was chosen as the application framework for **No Place Like Home** over other modern web frameworks (such as Ruby on Rails or ASP.NET) primarily because of the wide ecosystem of Python libraries and tools that exists today. Being able to easily integrate outside Python tools and libraries (like `pylucene`) greatly simplified development effort. Furthermore, Python was an obvious language choice because of the skill sets possessed by myself and researchers who are likely to continue this work.

The various features of the system are divided by functionality into a modularized architecture, with each system module being implemented as a Django application. Figure 3.6 displays these

³ <https://www.djangoproject.com/>

subsystems in the context of a deployment diagram.

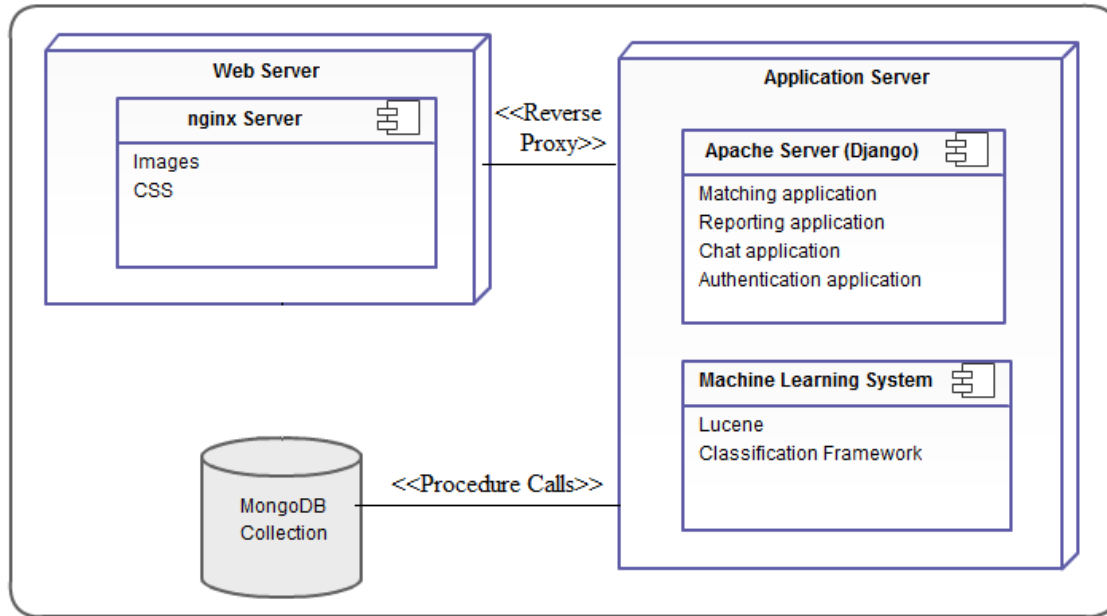


Figure 3.6: Deployment Diagram for **No Place Like Home**. Note that physical separation of the **nginx** server and the **Apache** server is optional.

Because the application must serve a large amount of static files (mostly in the form of pictures, see Figure 3.3), the primary web server which handles all direct requests is an **nginx**⁴ (“engine-X”) server. This web server is generally more efficient at serving static files to clients, and is used in this setup to serve all static data (cascading style sheets, images, etc.); a reverse proxy is set up on a specific URL (typically the root Django application URL, i.e., “/nplh/”) which forwards requests to an **Apache** server which hosts the Django installation via **mod_wsgi**.

The Django installation used is actually a fork of version 1.4 which supports the use of non-relational databases in the data layer of Django. The database driver used to support connectivity with MongoDB is an open-source driver known as **Django MongoDB Engine**, which provides object-relational mapping (ORM) for MongoDB collections as well as useful functionality like a map-reduce engine for common non-relational operations.

⁴ <http://www.nginx.org/>

Within the Django deployment are several core system applications: the **authentication**, **chat**, **reporting**, and **matching** subsystems. Although these applications make use of a common data store (described earlier in this chapter), they are loosely coupled and only refer to each other through URL redirection if necessary. Modularization of the architecture is an important design goal to reduce development and maintenance complexity.

The following sections describe the implementation of each of these components; each section describes the purpose and functionality of the component and, in addition, provides a description of its implementation by specifying the the Django views (in MVC terminology, the controllers) included in the component. In effect, these views describe the public interface of the applications.

authentication subsystem

The **authentication** subsystem includes functionality to handle all user-related tasks, such as user creation, authentication, linking accounts to existing social networks, importing social network data, and retrieving or editing user information. The majority of the views contained in this application deal with the process of user authentication and tie in to Django’s authentication mechanisms. This subsystem, to a large extent, incorporates an open source Django application called **Django-Social-Auth**⁵, which extends the default Django authentication system by supporting authentication via various third party providers. **No Place Like Home** utilizes the Twitter and Facebook authentication providers in order to both provide an efficient, easy-to-use account creation and authentication mechanism to users as well as harvesting relevant social connections from those accounts.

The other features included this application further extend this functionality by providing a custom user model and additional functions and views. The custom user model is necessary to reflect the **User** artifact in Django’s ORM by providing access to information like social reputation and a user’s associated work items. Other additions perform actions like retrieving a user’s social connections and editing a user profile. Table 3.3 details the views implemented in the

⁵ <https://github.com/omab/django-social-auth/>

`authentication` subsystem.

`chat` subsystem

The `chat` subsystem includes functionality which facilitates communication between digital volunteers via chat rooms. The backbone of this application is a third party Django application, **`django-jqchat`**⁶, which creates and manages chat rooms through standard HTTP interactions. The client is shown a chat control (see the lower portion of Figure 3.1) that is controlled and updated by simple asynchronous requests. Although **`django-jqchat`** performs the majority of the work in this application, additional functionality has been added to better manage the creation of chat rooms, their association with other artifacts in the system, and the history of content generated in a chat room. The two primary views (see Table 3.5) provided by this application's public interface are generally meant to be only called by asynchronous requests after initial page loads. Making regular requests at a specified interval keeps this process fairly lightweight. Using a solution which makes use of only Javascript is advantageous in that the server-side application code is integrated cleanly with the rest of the application, and in addition, the client is not burdened by a heavier implementation such as a Flash plugin.

Chat rooms are automatically created when a pet report is first selected by a user in the matching interface. This chat room is associated with the pet report and contains all the history of collaborative work on that report. Although this functionality lives in this application, it is actually utilized by the `matching` application.

The method by which contextual links (i.e., hypertext references that allow users to link to other artifacts in the system) are inserted into chat messages is implemented as an entirely client-side mechanism. Javascript allows the user to drag and drop various visual controls (such as pet reports) to various locations on the matching interface; controls also have a link that appears on mouseover which, when clicked, will insert a contextual link into the user's chat message. The link itself is then created via a standardized syntax that is transformed by additional Javascript into a hypertext link.

⁶ <http://code.google.com/p/django-jqchat/>

For example, after inserting a contextual link, the underlying string representation of a message might be, “Check out this dog: \$\$pet:714\$\$”. The client-side Javascript would transform the latter part of this message based on the syntax `pet:id` into the relative URL `/nplh/matching/pet/714/` with display text of “Pet-714”. Table 3.4 details the complete set of these transformations.

reporting subsystem

The **reporting** subsystem includes functionality to allow users to submit and edit lost or found pet reports. While this functionality caters to the data scout user, the editing component is open is more focused on the remote volunteer. The core views in this application are listed in Table 3.6.

This application, in addition to housing the primary data generation mechanisms for the system, also incorporates interaction with the machine learning system in the form of providing training examples for the **Pet** classifier (discussed in the next chapter). This interaction occurs when a new pet report is submitted (it becomes indexed in the information retrieval system) and when it is edited or upvoted. Upvoting a pet indicates a value judgment on the quality and completeness of that pet report (users are directed to make this kind of assessment based on provided instructions), and marks the pet report as a positive training example for the classifier. Editing a pet report indicates some kind of deficiency in the original report; the edited report is submitted as a positive training example. The original report can be compared with the newly edited report - the differences between them can be used as a negative training example.

matching subsystem

The **matching** subsystem is the heart of **No Place Like Home**. It facilitates the entire matching task work flow, which includes the two primary user interfaces in the system. In addition, the application also facilitates the verification work flow, which involves checkers voting on proposed matches. When a match receives enough votes, the contacts on each pet report in the match are notified to perform final verification; after this final verification, the pet reports are removed from

circulation. These two activities represent the bulk of the work done in the system (the only other major activity, the reporting of lost or found pets, being catered to by the **reporting** subsystem). Table 3.7 details the views contained in this subsystem.

It also features significant interactions with every other component of the system; in particular, it is the application which is the primary collaborator with the machine learning components of the system. When users select a pet to work on and enter the primary matching interface (see Figure 3.4 for a prototype of this interface), the **matching** application uses that pet as a query to the machine learning system. The machine learning system produces a ranked list of suggested matches which is used directly to output a selection of pets to the user. Furthermore, whenever a match is confirmed (and taken out of circulation), it is presented as training data to the **Match** classifier.

Machine Learning subsystem

The **Machine Learning** subsystem is not actually a Django application (i.e., it has no views) but instead acts as the public interface for accessing the machine learning components of the system. This system is more thoroughly examined in Chapter 4, but the public interface for the software is presented here. This interface presents functionality that allows the rest of the system to be agnostic to the underlying details of the machine learning components, which are the parts of this system that are most likely to change and evolve over time.

The public interface is focused on two types of activity: requesting a ranked list of pet matches for a given query (pet report) and providing an artifact to the machine learning system for retraining. The latter activity is interesting as it can be viewed as presenting observations of user activity to the machine learning components. Table 3.8 presents the public interface of this application in a similar fashion to the previous sections, but again, note that this application is not a Django application.

Table 3.3: This table details the views provided by the `authentication` application in terms of their arguments, HTTP method, purpose, and returned content.

Name (URL)	Method	Parameters	Purpose
<code>auth*</code> (<code>/login/</code> (<code>?P<backend></code>)/)	GET, POST	backend [Object containing third party authentication information.]	Primary authentication method; accepts a string <code>backend</code> that is used by a decorator to retrieve a third party service provider object. This method uses this object to redirect the user to the appropriate third party authentication page.
<code>complete*</code> (<code>/complete/</code> (<code>?P<backend></code>)/)	GET	backend [Object containing third party authentication information.]	This method is fired upon a successful return from a third party authentication page; it creates the internal Django user and various session cookies, authenticating the user.
<code>associate *</code> (<code>/associate/</code> (<code>?P<backend></code>)/)	GET, POST	backend [Object containing third party authentication information.]	Same as the <code>auth</code> view, but attempts to associate a third party account with the currently authenticated user.
<code>associate_-complete *</code> (<code>/associate/complete/</code> (<code>?P<backend></code>)/)	GET	backend [Object containing third party authentication information.]	This method is fired upon a successful return from a third party authentication page; it will associate the designated third party account with the current Django user.
<code>disconnect *</code> (<code>/disconnect/</code> (<code>?P<backend></code>)/)	GET	backend [Object containing third party authentication information.]	Disconnects (deauthenticates) the user from the given third party provider; this may result in logging the user out.
<code>error</code> (<code>/error/</code>)	GET	None	This method is fired when an error is encountered in the authentication process. Returns the user to a log in screen.
<code>profile</code> (<code>/profile/</code> (<code>?P<username></code>))	GET	username [string]	Retrieves the specified user's (the current user if none is specified) profile page.
<code>profile</code> (<code>/profile/</code>)	POST	Form Data [object describing a user profile]	Updates the current user's profile information with relevant form data from POST values.

*denotes functionality provided by the **django-social-auth** package.

URL format is the Django standard URL format (similar to regular expressions).

Table 3.4: This table lists the various textual transformations done via Javascript to support contextual links to system artifacts in the chat control.

Artifact	Syntax (Example)	Output URL	Output Text
Pet	<code>pet:ID</code> (<code>\$\$pet:714\$\$</code>)	<code>/nplh/matching/pet/(ID)</code>	Pet-(ID)
Match	<code>match:ID</code> (<code>\$\$match:26\$\$</code>)	<code>/nplh/matching/match/(ID)</code>	Match-(ID)
User	<code>user:Username</code> (<code>\$\$user:josh\$\$</code>)	<code>/nplh/auth/profile/(Username)</code>	Username

Table 3.5: This table details the views provided by the **chat** application in terms of their arguments, HTTP method, purpose, and returned content.

Name (URL)	Method	Parameters	Purpose
chat (/chat/(?P<id>)/ (?P<state>)/)	GET	id [int] state [int]	Retrieves the contents of the given chat room - if requested via an asynchronous call, the most recent contents (determined by comparing the provided state variable to the room's current state value) are returned via JSON message. If the user is not a member of this chat room, the user is added to the members of this chat room.
chat (/chat/ (?P<id>)/)	POST	id [int] Form Data [object containing chat message]	Called when a user posts a message to a chat room. The content of the chat room is updated and the room's state is incremented.

Table 3.6: This table details the views provided by the **reporting** application in terms of their arguments, HTTP method, purpose, and returned content.

Name (URL)	Method	Parameters	Purpose
report (/report/)	GET	state [int]	Displays a form allowing the user to submit a lost or found pet report, attaching relevant information (breed, color, description, etc.) and pictures.
report (/report/)	POST	Form Data [object describing pet report]	Validates and processes a submitted pet report, creating a Pet artifact if successful. Returns a confirmation page.
edit (/report/edit/ (?P<id>)/)	GET	id [int]	Retrieves the specified pet report and displays a form to the user which allows the user to edit the report or vote on the quality of the report.
edit (/report/edit/ (?P<id>)/)	POST	id [int] Form Data [object describing pet report]	Makes changes to the specified pet report using form data. Submits the changed report for training to the machine learning system.
upvote (/report/upvote/ (?P<id>)/)	POST	id [int]	Records an upvote from the current user for the specified pet report. Submits the report for training to the machine learning system.

Table 3.7: This table details the views provided by the `matching` application in terms of their arguments, HTTP method, purpose, and returned content.

Name (URL)	Method	Parameters	Purpose
<code>pet</code> (<code>/matching/pet/</code> (<code>?P<id>/</code>))	GET	id [int]	Displays a detailed view of the specified pet report. This view is generally used in AJAX calls to display a pop up form with the pet's information.
<code>match</code> (<code>/matching/match/</code> (<code>?P<id>/</code>))	GET	id [int]	Displays a detailed view of the specified match. This view is generally used in AJAX calls to display a pop up form with the match's information.
<code>vote_match</code> (<code>/matching/match/</code> (<code>?P<id>/</code>))	POST	id [int] Form Data [boolean for up or down vote]	Records the vote (positive or negative) of the current user for the specified match.
<code>home</code> (<code>/</code>)	GET	None	This view acts as the home page for the application; it displays the first screen of the matching interface, where a user selects a pet to work on.
<code>matches</code> (<code>/matching/matches/</code>)	GET	None	This view acts as the match verification page; returns a page containing all the current proposed matches.
<code>matching</code> (<code>/matching/(?P<id>/)</code>)	GET	id [int]	After selecting a pet to work on (from the detailed view), this view retrieves the information necessary to create the matching interface for that pet. This view interacts with the machine learning system to retrieve the suggested list of matches.
<code>propose_match</code> (<code>/matching/match/</code> (<code>?P<id_lost>/</code> (<code>?P<id_found>/</code>))	POST	id_lost [int] id_found [int]	Proposes a match between the two specified pet reports and creates a <code>Match</code> artifact.

Table 3.8: This table details the public interface of the **Machine Learning** subsystem. These are pure Python methods, and do not represent a Django application. The design and implementation of this system is described in the following chapter.

Method Name	Parameters	Purpose
<code>train_pet</code>	pet [Pet] positive [boolean]	Provides a Pet artifact as a training example to the Pet classifier. The boolean parameter indicates whether this is a positive or negative example.
<code>train_match</code>	match [Match] positive [boolean]	Provides a Match artifact as a training example to the Match classifier. The boolean parameter indicates whether this is a positive or negative example.
<code>query</code>	pet [Pet]	Queries the information retrieval and classifier systems for a list of pets that most closely correspond to the provided pet. These form a ranked list that is used in the matching subsystem.

Chapter 4

Machine Learning System Design and Evaluation

This chapter presents the design and architecture of the machine learning system used in **No Place Like Home**. This system is comprised of three components: an information retrieval system, a classifier for **Pet** artifacts, and a classifier for **Match** artifacts. The information retrieval system is implemented in **pylucene**¹, which is a Python wrapper around the popular Lucene search engine. The **Pet** and **Match** classifiers are designed to observe volunteer activity (in the form of utilizing the artifacts they produce) and render judgments on the respective artifacts they handle; the output of these classifiers is used to modify the rankings produced by querying the information retrieval system.

The public interface of this system, detailed in the previous chapter, is relatively simple and intended to allow consumers of this system to be very agnostic about its implementation. While this is, in general, a good software engineering principle to follow, the generic public interface is also important because the architecture of this system is still in many ways a prototype. The information retrieval system and classification framework are simple in design and intended to provide a baseline for both system design and the evaluation of the features used in these components. The concluding chapter of this work discusses future directions for development of this system.

The following sections discuss the design of the three primary system components and how they are combined to produce ranked query responses for the **matching** application. The remainder of the chapter is dedicated to detailing evaluations of the components individually and as a whole

¹ <http://lucene.apache.org/pylucene/>

in an experimental setting.

4.1 Design Considerations

The classifiers included in this system are designed to incorporate ongoing training as the activities of digital volunteers are observed. This is essential to the agential positioning of the machine computation performed by the system; it is key that the system adapts to new information coming from (human) collaborators over time. Because of this, the machine learning algorithms used in the system are supervised rather than unsupervised, meaning that they are trained using labeled data that act as training examples [6]. This is not to say that unsupervised algorithms are unsuitable for this domain, but simply that the supervised approach better synthesizes with the human activity taking place within the system.

Furthermore, because the system must adapt over time, the algorithms used must be capable of retraining and thereby adjusting their internal probabilistic models in response to new data (a process known as **active learning**). Essentially this means the classifier must either keep all of its training examples stored or be able to adjust its model using stored statistics (in the case of a probabilistic classifier). The latter method is preferred for performance reasons and can be accomplished in a variety of ways depending on the classification algorithm [6, 18]. In practice, **No Place Like Home** would likely perform active learning in batch processes (again, for performance reasons) rather than use a pure online approach to retraining.

Finally, because the classifiers are ultimately used to modify ranked retrieval results from an information retrieval system, and thus are ultimately not being used to make “hard” decisions [18], these classifiers have different requirements than traditional textual classification systems. Several environmental conditions play a significant role in the design and evaluation of these classifiers:

- The quality of the training data is not guaranteed to be high for many reasons:
 - * Data generation in the system is relatively unrestricted and the correctness of a pet report, in terms of grammar, spelling, and adherence to format (i.e., putting only breed

information in a “breed” field) cannot be assured.

- * Although digital volunteers can improve poor pet reports by editing them, they are not directed, trained annotators and thus training examples cannot be expected to have consistent characteristics across their entire population.
- * The nature of the problem domain makes textual classification inherently difficult (see Chapter 2).
- The central problem in matching pets is that there is **at most** one correct pet in the system. This generally means that the classifiers in the system must favor **recall** (or **sensitivity** in this context) over **precision**; it is paramount that the classifier correctly categorize the true positive, if it exists. False positives can be forgiven easily as long as the true positive improves in terms of its search ranking. Again, we can rely on human computation (and subsequent verification by the crowd) to solve the hard problem as long as the machine learning components are helping to reduce the problem of large search spaces.

4.2 Lucene Index

The core component of the machine learning systems is, somewhat misleadingly, actually an information retrieval system. The **matching** application must present a ranked list of suggested pets to a user in response to a selected pet; this process is directly analogous to querying a search engine for a list of relevant documents (with the caveat that in this system there is at most one relevant document to any query). Using this analogy, **Pet** artifacts can easily be viewed as multi-field documents to be indexed in an information retrieval system.

As mentioned in this chapter’s introduction, **No Place Like Home** implements this information retrieval system as a Lucene index. **Pet** artifacts are translated to multi-field documents (i.e., the **breed** property on a **Pet** artifact is translated into a corresponding field in the Lucene document). Lucene processes these documents (using Lucene’s **SimpleAnalyzer** tokenization method) and stores relevant information about the documents and their terms, like **tf · idf** scores.

When the **matching** application presents a **Pet** artifact to this subsystem, the supplied pet report is translated into a Lucene query using multi-field query syntax. The only hard matching requirement set on this query is for the **lost** field, which indicates whether or not the report is for a lost or found pet (this value is set to the opposite of the supplied artifact’s value). The query is executed using Lucene’s standard searching tools (**IndexSearcher**) and a ranked list of documents (pet reports) is returned. This ranked list is then reordered according to the scoring formula (describe later in this chapter) to obtain a final ranking which is then returned to the **matching** application.

4.3 Pet Classifier Architecture

The objective of the **Pet** classifier is to obtain some measure of the quality and completeness of a given pet report. This information is assumed to be valuable because a pet report that is incomplete or of poor quality is less likely to be matched in general than one that is of high quality and fully described.

This component is designed as a simple binary classifier which labels a **Pet** artifact as either positive or negative. A positive label indicates that the pet report is of high quality and completeness, and a negative label indicates the opposite. In practice, the scoring function utilizes the probability that the **Pet** artifact belongs to the positive class:

$$p(C_1|F) = \frac{p(C_1)p(F|C_1)}{p(F)}$$

where F is a feature vector representing the pet report, and C_1 is the positive class.

This classifier is Bayesian, as can be seen in the formula above. The feature vector representation of a pet report is implemented as a bag of words, with each term in the vocabulary represented by a particular index in the one-dimensional vector; this is a standard model used in information retrieval and text classification [6, 12].

One feature of the **Pet** artifact does not directly translate to a binary feature — this is the **revisions** attribute, which tracks user judgments and edits to a pet report, and is intuitively

the most important feature for this classifier. We can translate this feature into numeric form by counting the number of revisions with no edits (which act as upvotes to the pet report). This number can be made even more interesting when multiplied by the social capital of the users involved (and properly normalized), making its potential value as a distinguishing feature even greater.

Still, this number is not compatible with a bag-of-words feature vector. The simplest way to resolve this problem is to assign a distinct token in the vocabulary for each occurring value of this number, but this approach is problematic. A better approach is to stratify the possible values of this number into distinct tokens in the vocabulary [10]. For example, a numeric value ranging between 0 and 100 might be discretized into four separate tokens, “**zero**, **over10**, **over30**, **over70**”. This approach has the advantage in that the tokens created can be more meaningfully selected and result in information gain.

I have thus far neglected to mention the specific classification algorithm that this classifier implements. This is because, in the interest of exploration, I have evaluated various algorithms for this classifier. These variations include two Naive Bayes classifiers (multinomial and Bernoulli), a Support Vector Machine (SVM), a Nearest Neighbors classifier, and a Decision Tree classifier. The classifier described above fits the description of a Naive Bayes classifier because these classifiers best fit the design requirements described in this chapter; they can be designed for active learning, and in addition, they exhibit high bias, which performs well with the relatively small amounts of data present in the system. I will discuss the evaluation of these various algorithms later in the chapter.

4.4 Match Classifier Architecture

Like the **Pet** classifier, the **Match** classifier is implemented as a Bayesian classifier (though, like the **Pet** classifier, I evaluate several different classification algorithms for this component). The goal of this classifier is to examine a pair of pet reports and render a judgment about whether or not the pair constitutes a likely match or not. This information is naturally assumed to be valuable to the process of ranking potential matches for the **matching** application. In many ways this classifier acts as a collaborative filter, using successful matches to try to discover other potential matches.

Unlike the **Pet** classifier, the **Match** classifier does not directly include textual tokens into its feature vector. Instead, the feature vector examines the similarity of the two pet reports on a field by field basis. The specific information in each pet report which is considered is breed, color, size, sex, age, and description. Before comparison, the strings contained in each field are cleaned, removing punctuation and character case. After this, the fields are then tokenized and placed into a set to facilitate comparison. Each field comparison results in two features - whether or not the corresponding fields are disjoint, and the number of overlapping tokens in each field, normalized by field length, which results in the percentage of overlapping tokens. This latter feature is then stratified like the **revisions** attribute was for the **Pet** classifier, using the same methodology [10]. This stratification results in six separate features. Combined with the feature which categorizes disjointness, this entire process creates a feature vector with 42 binary features.

It is notable that this classifier does not actually consider the content of the pet reports, as does the **Pet** classifier. This is by design; the training data for matches is extremely limited, rendering content classification less useful. Furthermore, the content of pet reports is, intuitively, not actually that useful to determining whether or not they constitute a good match. For example, two similar reports about a labrador retriever are not a good potential match because they are about labrador retrievers; they are a good potential match because they are both about the same breed of animal. By considering only the similarity of the fields between the two pet reports, this classifier seeks to discover what the most useful correlations are within individual pet reports.

4.5 Combined Scoring System

Using the three components of this system, I define a combined scoring system that produces the rank of a given pet report in relation to another pet report. In practice, the system uses this scoring system to reorder the ranked results of a Lucene query, as discussed previously. The most complete scoring algorithm (in the sense that it utilizes all the components of the system) is as follows:

$$S(P_1, P_2) = L(P_1, P_2) \times \frac{C_P(P_1) + C_P(P_2)}{2} \times C_M(P_1, P_2) \quad (4.1)$$

where P_1 is the query pet report, P_2 is a potential matching pet report, $L(P_1, P_2)$ is the original Lucene score for the pair, $C_P(x)$ is the **Pet** classifier’s probability estimate that x belongs to the positive class, and $C_M(P_1, P_2)$ is the **Match** classifier’s probability estimate that the pair is a match.

It should be noted that the probability estimates given by each classifier are true probability estimates because of the need for normalization (probability estimates produced by classifiers are usually just scores because there is often no need for this normalization if the classifier is used independently).

Formula 4.1 utilizes all possible components of the system. However, because the evaluation of this system is exploratory, I have evaluated several variations of this scoring formula for the combined evaluation of the system discussed at the end of this chapter. These variations are described by Table 4.1.

4.6 Experimentation and Evaluation

The evaluation of this system is two-fold, incorporating both individual evaluations of each classifier as well as a combined evaluation which examines how well the scoring formulas (described in the previous section) improve the rankings of true positive matches in ranked results. The following sections describe the structure of these evaluations, discussing the corpus and methods used to obtain labeled training data, as well as the specific implementation details of various classification algorithms.

Evaluation of both the individual classifiers and the combined system was always performed through 10-fold cross validation. Experimental results for the individual classifiers are presented in terms of these folds, as well as aggregated metrics. Although fold results are presented in terms of accuracy figures, the more important metrics are the overall precision, recall, and F_2 score for the classifiers. Because of the emphasis on recall, I chose to calculate the F_2 measure, which is a

variation of the harmonic mean of precision and recall which weights recall higher than precision.

4.6.1 Corpus

The problem domain of pet-to-family reunification is very different from traditional textual classification domains; finding an appropriate corpus of data was difficult. Many online services which index reports of lost pets do not also index reports of found pets, which have semantic differences. Ultimately I used publicly available data obtained from “America’s National Lost & Found Pet Database”² (operated by the Alabama Pet Registry) as a raw corpus of lost and found pets.


Pet ID:	pet type: dog	breed: chihuahua	
sex: male	size: small	color: medium tan	age: 18-24 months
date lost: Feb 29, 2012			
location where lost:			
other info: has a cherry eye, friendly, small			
 (click image to enlarge)			

Figure 4.1: Sample pet obtained from corpus.

Figure 4.1 displays a (censored) sample entry from this database. Entries in this database contain information about pet type, breed, sex, size, color, age, date lost or found, location lost or found, a field for “other information”, and one or more pictures of the pet. Although interesting (and possibly useful for future work), I removed information regarding location and date as the pets retrieved from the database were spread geographically across the United States and wide range of dates, which is not realistic for a disaster event. Furthermore, the “other information” field varied widely in terms of its content, ranging from being empty to containing multiple paragraphs of grammatically incorrect, irrelevant information.

I immediately removed from my corpus those pets which did not have pictures attached to their reports. This cut down the size of the corpus from just over 2000 reports to just over 800

² <http://www.lostfoundpets.us/>

reports. The distribution between lost and found pet reports was approximately even, so I was able to leave this initial corpus fairly in tact barring the changes discussed above.

4.6.2 Classifier Implementation

Classifiers were implemented in Python using the **scikit-learn** machine learning library [17]. This library supports many classification algorithms and offers support for evaluation and experimentation. Table 4.2 details the various classification algorithms used in evaluating the **Pet** and **Match** classifiers.

4.6.3 Pet Classifier Training

To obtain labeled training examples of complete, high-quality and incomplete, low-quality pet reports, I constructed an annotation application to allow volunteers to evaluate and edit pet reports in the corpus. Ten volunteers, selected from a convenience sample, assisted in this annotation process. This application’s interface is shown in Figure 4.2.

The annotation application asks volunteers to render their judgment on whether or not the pet report is complete and informative and provides brief annotation instructions, such as checking the spelling of the content of the pet report. If the volunteer feels the pet report is of high quality, they can register an upvote for that report. If the report is lacking, the volunteer can edit the pet report and submit a new revision of it. After submitting an upvote or a revision, the annotation application would display a new pet report at random to the user.

The majority of pet reports in the original corpus received revisions rather than upvotes; these revisions usually took the form of either spelling corrections or major changes to the description field. Annotators were instructed that this field should contain a concise, relevant description of the animal that encompassed features not captured by the other fields. For example, a good description might be “Dog has white chest and large black spot on back left leg; long hair.”

Because this field was initially populated with the “other information” field obtained from the raw corpus, however, it often contained information that was irrelevant to the pet, such as contact


no place like home
PROJECT EPIC
MATCH PETS
VERIFY MATCHES
SUBMIT LOST/FOUND PET
MY PROFILE
ABOUT

Is this pet report complete and informative?

Please review the pet report below to determine its quality. Review with consideration to quantity of information (are all the fields filled in?), quality of information (is the picture relevant? is the description free of fluff and to the point?), and general correctness (is everything spelled correctly?). If the pet report meets these criteria, click the link below. If it needs changes to meet the criteria, make these changes by editing the report below and then clicking the "Submit Changes" button at the bottom of this page.

[YES, this pet report is complete](#)

(if not, edit it below and then submit your changes!)



Pet Details:
Image is useful: ☒ Yes ☐ No
(if the image to the left is relevant to this report, select "Yes", otherwise select "No")
Lost / Found: **Found**
Breed:
Color:
Size:
Age:
Description:
(a relevant description of the animal, i.e., "large black spot on left leg", not "cutest dog ever!")

Figure 4.2: Interface allowing volunteers to edit or upvote pet reports (this interface is utilizing the reporting application).

information, or sentimental text. Although both of these have a place in **No Place Like Home**, these fields obscure the classification process and annotators removed this irrelevant information.

Pet reports that were upvoted became positive training examples. If a revision was made to a pet report by an annotator, the previous version of the pet report was used as a negative training example (unless that report had upvotes, which was a rare case), with the revised version being used as a positive training example.

This process resulted in over 1000 positive training examples and over 800 negative training examples being generated for use in classification.

4.6.4 Pet Classifier Evaluation

The labeled data generated by annotators was used in 10-fold cross validation to train and evaluate the various implementations (see Table 4.2) of the **Pet** classifier. Table 4.3 displays the results of this evaluation.

The Bernoulli Naive Bayes classifier outperforms the other classifiers in terms of accuracy, precision, and F_2 score, losing the accuracy title in only two folds to the Nearest Neighbors and Decision Tree. Overall, however, it is unsurprising that the Naive Bayes classifiers outperformed the Nearest Neighbors and Decision Tree classifiers. As discussed before, Naive Bayes classifiers are well suited to the type and quantity of training data present in this evaluation.

4.6.5 Match Classifier Training

To obtain labeled training examples of pet matches, I extended the existing annotation application to allow users to submit pet reports which would serve as matches to pet reports within the system. Although I had originally planned to use matches that existed in the metadata of the original corpus, I found these matches to be unsuitable for training and evaluation because they consisted largely of junk data. I therefore set out to create synthetic ground truth data in a manner that would preserve as much ecological validity as possible.

To do this I identified lost pet reports that had more than one picture associated with their

report. The annotation application used to train the **Pet** classifier only uses the first picture associated with a report; the other pictures associated with pet reports had not been seen previously by annotators.

I extended the annotation application with another interface that displays an alternate picture of a pet belonging to the aforementioned set and asks the user to submit a found pet report for that pet, using only the picture (see Figure 4.3). This task mirrors the actual problem facing a data scout who takes pictures of pets in a shelter or finds a pet and must describe them using (potentially) only visual information. In fact, the data scout is at an advantage because he or she can more easily determine the size, sex, and visual characteristics of the animal in question because he or she presumably does not need to rely only on a photograph of the animal. The annotators, unfortunately, do not possess this advantage.

Figure 4.3: Interface allowing volunteers to submit a found pet report which corresponds to an existing lost pet report. The application displays an alternate picture from the original pet report.

Annotators generated 101 “matches” using this interface. The size of this set in proportion to the data set as a whole roughly corresponds to match percentages seen in real life systems (5% to 15% - see Chapter 2). These matches were all used as positive training examples. Negative training

examples were created by generating a set of random matches between other pet reports outside the positive set.

4.6.6 Match Classifier Evaluation

The labeled data generated by annotators was used in 10-fold cross validation to train and evaluate the various implementations (see Table 4.2) of the `Match` classifier. Table 4.4 displays the results of this evaluation.

The accuracy metric for this evaluation is misleading in that it closely tracks the class prior probabilities (i.e., $p(C_{match}) \approx .1$). This explains the haphazard winners of the accuracy metric across the various folds. Again, there is good reason to believe that the most useful evaluation metric for classification in this domain is recall; here the Naive Bayes classifiers outperform the other classifiers.

Both versions of Naive Bayes perform essentially identically; this is to be expected because feature vectors provided to the `Match` classifier are always binary-valued. In practice, however, the Bernoulli Naive Bayes classifier seems best suited to this classification task.

4.6.7 Combined Evaluation

In each evaluation the Bernoulli Naive Bayes implementation of each classifier outperformed the other algorithms (in terms of recall and F_2 score) and thus was selected as the implementation used for each classifier in the combined evaluation, which is essentially an end-to-end evaluation of the machine learning system. This evaluation is meant to discover the effectiveness of the two classification components and how best to combine them with the results of the Lucene scorer. This evaluation examines all the scoring algorithms in Table 4.1.

The baseline metric in this evaluation is the average and median ranking of the true positive across the set of all annotator-created matches as produced by Lucene. The effectiveness of the various scoring functions (and consequently, that of the classification components as well) is judged by how well the average and median ranking of the true positives is improved (reduced) after

re-ranking is completed. Table 4.5 displays the results of this final evaluation.

Although the baseline metric was calculated across the entire set of positive matches using just Lucene, it is important to note that 10-fold cross validation was again employed in evaluating the various scoring functions (to ensure that classifiers were never to used re-rank queries whose constituent **Pet** artifacts were members of their training sets). The recalculated rank statistics displayed in Table 4.5 are the averages across the folds. Furthermore, the **Match** classifier uses the same set of negative training examples for this evaluation as it did in its individual evaluation to ensure consistency.

The first takeaway from these results is that generic, baseline Lucene query rankings perform fairly well to begin with; the baseline median rank of the true positive is 26. It is important to remember that the result set size in this evaluation is fairly small (the median size being 235 results). Furthermore, it seems useful to consider the median rank with higher regard than the mean rank, as a few outliers in the 101 queries skew the mean statistic. Finally, I do not provide precision metrics for this ranked retrieval evaluation because it is fairly meaningless (there is at most one relevant document for any query). However, if in the future (after ongoing interface design work is complete) an estimate for the number of pet reports that could be displayed at one time on a user’s web browser became available, it would be interesting to see a precision-at- k metric to evaluate whether or not the system is doing a good job at getting the true positive “above the fold”.

It is apparent from the results of this end-to-end evaluation that the **Pet** classifier was generally unhelpful in improving the rank of the true positive. This could be due to a variety of reasons; the classifier’s performance was not high enough, or the difference between the classes (complete and informative vs. incomplete and uninformative) is not pronounced enough in the labeled training data produced by volunteers. Finally, it could be that regardless of the performance of the classifier, this information is actually not relevant to whether or not two pet reports are a good match - this would invalidate the hypothesis which is the reason for the development of this classifier. It is interesting to note, however, that the negative impact of this classifier on the re-ranking process was greatly mitigated when combined with the results of the **Match** classifier (in scoring formula S_1).

The `Match` seems to have performed admirably in this evaluation despite displaying average performance in its individual evaluation. The classifier appears to be able to correctly categorize the true positive in the rankings with high probability estimate (boosting the ranking of the true positive) and does a good enough job with the other results that the ranking of the true positive across the query set is improved. Again, we see that the mean ranking is skewed by outliers, but the median rank shows significant improvement, with the S_4 scoring function resulting in a 61.5% improvement in median rank of the true positive.

This result is the most promising presented in this chapter. It is tempting to view the individual classification evaluation results and even these end-to-end evaluation results as average at best, but this is to be expected; in Chapter 2 I discussed the many challenges that this machine learning system faces. In assessing these results, it is important to note that any improvement in the end-to-end evaluation is good, because the machine learning system is not expected to make hard decisions about what combinations of pet reports are or are not viable matches. The system relies on the foundation of human computation to make this hard decision; its role is only to assist in this decision.

Table 4.1: This table details the variations of the general scoring formula which are used in the combined evaluation of this system.

Name	Equation	Description
S_1	$S_1(P_1, P_2) = L(P_1, P_2) \times \frac{C_P(P_1) + C_P(P_2)}{2} \times C_M(P_1, P_2)$	Identical to the general formula; incorporates all components of the system.
S_2	$S_2(P_1, P_2) = L(P_1, P_2) \times \frac{C_P(P_1) + C_P(P_2)}{2}$	Does not utilize the Match classifier.
S_3	$S_3(P_1, P_2) = L(P_1, P_2) \times C_P(P_2)$	Does not utilize the Match classifier or the Pet classifier's score for P_1 .
S_4	$S_4(P_1, P_2) = L(P_1, P_2) \times C_M(P_1, P_2)$	Does not utilize the Pet classifier.

Table 4.2: This table details the various classification algorithms that were used as implementations for the **Pet** and **Match** classifiers.

ID	Algorithm	Notes
MNB	Multinomial Naive Bayes	Variant of Naive Bayes that permits non-binary features (preserves term counts for the Pet classifier).
BNB	Bernoulli Naive Bayes	Assumes features are binary-valued; binarizes feature vectors (discards term counts for the Pet classifier).
KNN	K-Nearest Neighbors	k defaults to 5.
DT	Decision Tree	Utilizes Gini impurity as the splitting algorithm.

Table 4.3: This table displays the evaluation results for each implementation of the **Pet** classifier. The displayed statistic for each fold is the accuracy of that classifier for that fold. Overall accuracy, precision, recall, and F_2 scores are also provided. **Bolded** cells indicate the best-performing classifier for that metric.

	Folds										Mean Evaluation Metrics			
ID	1	2	3	4	5	6	7	8	9	10	Acc.	Prec.	Rec.	F_2
MNB	.639	.707	.611	.615	.615	.716	.668	.553	.611	.659	.639	.524	.907	.789
BNB	.678	.707	.615	.635	.625	.736	.678	.563	.625	.682	.654	.536	.900	.790
KNN	.534	.481	.635	.582	.558	.529	.615	.529	.615	.530	.561	.400	.242	.262
DT	.591	.606	.567	.620	.587	.582	.601	.625	.538	.580	.590	.477	.504	.497

Table 4.4: This table displays the evaluation results for each implementation of the **Match** classifier. The displayed statistic for each fold is the accuracy of that classifier for that fold. Overall accuracy, precision, recall, and F_2 scores are also provided. **Bolded** cells indicate the best-performing classifier for that metric.

	Folds										Mean Evaluation Metrics			
ID	1	2	3	4	5	6	7	8	9	10	Acc.	Prec.	Rec.	F_2
MNB	.877	.913	.864	.930	.852	.877	.901	.951	.901	.933	.900	.598	.598	.592
BNB	.877	.914	.864	.926	.852	.877	.901	.951	.901	.933	.900	.598	.598	.592
KNN	.864	.852	.914	.923	.914	.926	.901	.963	.901	.921	.908	.710	.412	.444
DT	.901	.889	.938	.901	.901	.889	.937	.951	.901	.933	.914	.702	.541	.564

Table 4.5: This table displays the evaluation results for the combined, end-to-end evaluation of the various scoring functions and their components. The rank figures shown are the mean and median ranks of the true positive across the result sets. Percentage improvement is calculated using a result set size of 235, which was equal to the median result set size of all 101 queries (queries at most return half of the index due to the hard querying requirement splitting lost and found pet reports).

Scoring Function	Mean Rank	Median Rank	% Improvement to Mean Rank	% Improvement to Median Rank
Lucene Baseline	54.32	26	N/A	N/A
S_1^*	54.65	13	-0.61%	50%
S_2^*	88.9	48	-63.7%	-84.6%
S_3^*	107.5	65	-97.9%	-150%
S_4^*	49.39	10	9.1%	61.5%

*Results shown are the averages across 10-fold cross validation.

Chapter 5

Conclusion

This chapter reviews the work presented in this thesis, summarizing the key points from each chapter, reflecting on the results of this work and its contributions, and discussing future work.

5.1 Summary

Chapter 2 explored the background behind this work, discussing the problem domain of pet-to-family reunification as well as relevant literature which informs the rest of the thesis. Displaced pets are a large problem in disaster situations, giving rise to various public safety and health issues. Furthermore, single individuals are ill-suited to the task of locating a specific pet for a variety of reasons, the most prominent being the potential size of the search space. Utilizing digital volunteerism to facilitate crowd work to help address this problem mitigates this problem; creating a specialized system for digital volunteers to perform this work mitigates some of the challenges facing digital volunteers. There is also space for machine learning systems to aid in the completion of this task. Because the task is inherently difficult, the approach taken in this work integrates human computation with machine computation, with the latter assuming the paradigm of an agent collaborating with users.

Chapter 3 presents the system design and architecture of **No Place Like Home**, a software platform for facilitating pet-to-family reunification. The system's interface and architectural design are influenced primarily by the envisioned tasks of volunteers: reporting pets, matching lost pet reports with found pet reports, editing pet reports, and verifying proposed matches. Motivated by

a diverse user demographic and the need to provide a simple yet powerful interface for volunteers to complete these tasks, a key feature of this system is an accessible user interface, which is an area of ongoing collaborative work [1]. Collaboration mechanisms and social capital constitute the other key design features of **No Place Like Home**. The software architecture of the system is a standard client-server model; data is stored in a document-oriented store, and the software features of the application are implemented as Model-View-Controller Django applications.

Chapter 4 discusses the design and evaluation of potential implementations of the machine learning components of **No Place Like Home**. An information retrieval system, implemented in Lucene, is used as a backbone for performing ranked retrieval of pet reports corresponding to a given pet, which is the essential functionality used by the `matching` application. Two classifiers are designed and evaluated; the first is the `Pet` classifier, which attempts to model the quality of a pet report, and the second is the `Match` classifier, which attempts to model the correlations between successful pet matches. Various implementations of these classifiers (Naive Bayes, Nearest Neighbors, and Decision Tree algorithms) are individually evaluated. The labeled data used for each evaluation was generated by annotators working off of a raw corpus (obtained from a public database of lost and found pet reports), performing annotation tasks that closely mirror predicted digital volunteer activities. The evaluation of each classifier revealed that Naive Bayes implementations performed the best. In an end-to-end evaluation of the system, various scoring functions (which incorporated the two classifiers to different extents) were used to re-rank baseline Lucene rankings generated for 101 queries that included a true positive (i.e., the query was a pet that had a matching report generated by an annotator). This evaluation showed that the `Pet` classifier was generally ineffective at providing information useful to re-ranking results and identifying the true positive match, whereas the `Match` classifier was effective at improving the rank of the true positive over the result set.

5.2 Reflection and Contributions

One of the primary contributions of this work is the design and architecture of **No Place Like Home**; this effort is informing the ongoing implementation of the system in preparation for a public deployment in a real disaster setting. Furthermore, the evaluations performed in Chapter 4 are valuable in assessing the worth of the approach taken in this work, which is to combine human computation and machine computation in a symbiotic manner to better approach the problem of pet-to-family reunification. Although the original scoring system, which incorporates machine learning measures of pet report quality and match likelihood (the **Pet** and **Match** classifiers, respectively), was relatively unsuccessful, the other explorations performed in this work reveal that even a simple Lucene implementation combined with the **Match** classifier hold great promise in fulfilling the role of a machine learning collaborator which observes and supports the human computation being done by digital volunteers as a crowd.

5.3 Future Work

The most apparent need for future work on this system is a public deployment of the system in an actual disaster scenario. This would permit the study of the various social mechanisms proposed as system features in addition to providing an opportunity to study the machine learning components of the system in a more realistic setting. Furthermore, a system deployment would permit the design and evaluation of potentially interesting features that could be included in the machine learning components of the system but were infeasible to simulate and evaluate in the experimental setting presented in Chapter 4. For example, it would be interesting to consider location correlations between pet reports (I envision this being done by using some kind of stratification via clustering), or correlation between the sets of users who have submitted or worked on pet reports as valuable features for the **Match** classifier.

Furthermore, more work is needed to continue to develop the machine learning components of the system in general. The exploratory implementations and evaluations of the classification

algorithms presented in Chapter 4 are simply platforms for future iteration and refinement. In addition, the core information retrieval system used in Chapter 4 was deliberately simple in its implementation (because the focus of the evaluation was on the classification components); more work in tuning this component would undoubtedly result in higher performance. Even radically different components could be considered for future development of the machine learning system; the primary purpose of this work's implementations and evaluations was to assess the potential improvements that a machine learning system can offer to the problem of pet-to-family reunification.

Bibliography

- [1] Mario Barrenechea, Joshua Barron, and Joanne White. No Place Like Home: Pet-to-family reunification after disaster. In Proceedings of the 2012 annual conference extended abstracts on Human factors in computing systems (to appear), CHI EA '12, New York, NY, USA, 2012. ACM.
- [2] William J. Corvey, Sarah Vieweg, Travis Rood, and Martha Palmer. Twitter in mass emergency: what NLP techniques can contribute. In Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media, WSA '10, pages 23–24, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [3] Martin Fowler. Patterns of Enterprise Application Architecture. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [4] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, ECAI '96, pages 21–35, London, UK, UK, 1997. Springer-Verlag.
- [5] C. E. Fritz and J. H. Mathewson. Convergence behavior in disasters: A problem in social control. Committee on Disaster Studies, National Academy of Sciences, National Research Council, Washington DC, 1957.
- [6] Daniel Jurafsky and James H. Martin. Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.
- [7] J. C. R. Licklider. Man-computer symbiosis. IEEE Ann. Hist. Comput., 14(1), January 1992.
- [8] Henry Lieberman. Autonomous interface agents. In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '97, pages 67–74, New York, NY, USA, 1997. ACM.
- [9] S.R. Lowe, J. Rhodes, L. Zweibach, and C. Chan. The impact of pet loss on the perceived social support and psychological distress of hurricane survivors. Journal of Traumatic Stress, 22(3):244–247, 2009.
- [10] Sofus A. Macskassy, Haym Hirsh, Arunava Banerjee, and Aynur A. Dayanik. Using text classifiers for numerical classification. In Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2, IJCAI'01, pages 885–890, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

- [11] Pattie Maes. Agents that reduce work and information overload. Commun. ACM, 37(7):30–40, July 1994.
- [12] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA, 2008.
- [13] Megan McNabb. Pets in the eye of the storm: Hurricane Katrina floods the courts with pet custody disputes. Animal Law, 14(71), 2007.
- [14] Leysia Palen. Datascouts: Citizens as information gathers for crisis event reconnaissance activities.
- [15] Leysia Palen, Kenneth M. Anderson, Gloria Mark, James Martin, Douglas Sicker, Martha Palmer, and Dirk Grunwald. A vision for technology-mediated support for public participation & assistance in mass emergencies & disasters. In Proceedings of the 2010 ACM-BCS Visions of Computer Science Conference, ACM-BCS '10, pages 8:1–8:12, Swinton, UK, UK, 2010. British Computer Society.
- [16] Leysia Palen and Sophia B. Liu. Citizen communications in crisis: anticipating a future of ICT-supported public participation. In Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '07, pages 727–736, New York, NY, USA, 2007. ACM.
- [17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay E. Scikit-learn: Machine Learning in Python . Journal of Machine Learning Research, 12:2825–2830, 2011.
- [18] Fabrizio Sebastiani. Machine learning in automated text categorization. ACM Comput. Surv., 34(1):1–47, March 2002.
- [19] Kate Starbird. Crowd computation: organizing information during mass disruption events. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion, CSCW '12, pages 339–342, New York, NY, USA, 2012. ACM.
- [20] Kate Starbird and Leysia Palen. Voluntweeters: self-organizing by digital volunteers in times of crisis. In Proceedings of the 2011 annual conference on Human factors in computing systems, CHI '11, pages 1071–1080, New York, NY, USA, 2011. ACM.
- [21] Kate Starbird and Jeannie Stamberger. Tweak the tweet: Leveraging microblogging proliferation with a prescriptive grammar to support citizen reporting. In Proceedings of the 2010 Information Systems for Crisis Response and Management Conference, ISCRAM 2010, Seattle, WA, USA, 2010.
- [22] Sudha Verma, Sarah Vieweg, Will Corvey, Leysia Palen, Jim Martin, Martha Palmer, Aaron Schram, and Kenneth M. Anderson. Natural language processing to the rescue? Extracting “Situational Awareness” tweets during mass emergency. In the Fifth International AAAI Conference on Weblogs and Social Media, 2011.